软件回归测试报告

基于Flask的深度学习自动化部署系统

|  |  |  |  |
| --- | --- | --- | --- |
| 编写 | 许京爽，崔昕宇，张利鹏，聂磊 | 日期 | 2020年5月24日 |
| 校对 | 聂磊 | 日期 | 2020年5月24日 |

北京航空航天大学 计算机学院

软件工程综合实验课 C组

二〇二〇年五月二十四日

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 |
| 1.0.0 | 2020.5.24 |  | 完成第一版 | 崔昕宇，张利鹏，许京爽，聂磊 |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

[1 简介 1](#_Toc41342556)

[1.1 目的 1](#_Toc41342557)

[1.2 测试用例的维护方式 1](#_Toc41342558)

[1.3 回归测试策略 2](#_Toc41342559)

[1.4 回归测试的基本过程 2](#_Toc41342560)

[2 基线测试用例的回归测试结果 3](#_Toc41342561)

[2.1 代码单元的回归测试 3](#_Toc41342562)

[2.2 功能需求的回归测试 4](#_Toc41342563)

[2.3 软件质量特征的回归测试 6](#_Toc41342564)

[3 针对评审意见新增的测试用例 6](#_Toc41342565)

[3.1 用例501 未登录用户越权访问用户数据的安全测试 6](#_Toc41342566)

[3.2 用例502 配置文件与数据库信息同步动作的测试 7](#_Toc41342567)

[4 总结与反思 7](#_Toc41342568)

[参考文献 7](#_Toc41342569)

基于Flask的深度学习自动化部署系统

回归测试报告

**关键词：**深度学习，数据即服务（DaaS）

**摘要：**本文描述了2020春季学期《软件工程综合实验》课程中，C组成员开发的深度学习自动化部署系统的回归测试报告。

**专业术语清单：**

|  |  |
| --- | --- |
| **专业术语** | **中文解释** |
| SRS | SRS指软件需求规格说明书（Software Requirements Specification），是软件需求设计阶段的产物，描述了软件所需满足的需求。 |
| Python | Python是一种跨平台的计算机程序设计语言，具有面向对象和动态类型的特性。Python最初被设计用于编写自动化脚本，随着版本的不断更新和语言新功能的添加，越多被用于独立的、大型项目的开发。 |
| URL | 统一资源定位系统（Uniform Resource Locator）是因特网的万维网服务程序上用于指定信息位置的表示方法。 |
| 深度学习 | 深度学习是机器学习领域中一个新的研究方向，通过学习样本数据的内在规律和表示层次，让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。 |
| 模型部署 | 将训练完毕的深度学习模型放置在特定硬件平台上，辅以合适的软件环境，使该模型能够被用户使用。 |
| HTTP | 超文本传输协议（英文全称：HyperText Transfer Protocol）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础。 |
| REST API | 表现层状态转换（英文全称：Representational State Transfer）是一种万维网软件架构风格，目的是便于不同软件/程序在网络中互相传递信息。表现层状态转换是根基于超文本传输协议之上而确定的一组约束和属性，是一种设计提供万维网络服务的软件构建风格。 |

# 1 简介

## 1.1 目的

本文档是C组开发项目 ***基于Flask的深度学习自动化部署系统*** 的回归测试报告，在完成软件缺陷修复之后，本组对系统进行了回归测试，以确保软件缺陷修复过程没有引入新的程序错误。全文内容包含了构建软件测试用例集的策略，回归测试的执行过程，以及回归测试用例与测试结果。

本文档是为软件开发人员、测试人员、项目验收方编写的，可作为测试过程的评价指标和项目验收达标的依据。

## 1.2 测试用例的维护方式

软件的测试人员在实施测试的过程中将所设计的测试用例保存到测试用例集中，并对其进行维护和管理。当得到一个软件的基线版本时，用于基线版本测试的所有测试用例就形成了基线测试用例集。在需要进行回归测试的时候，可以根据所选择的回归测试策略，从基线测试用例集中提取合适的测试用例组成回归测试用例集，通过运行该集合中的测试用例来实现回归测试。保存在基线测试用例集中的测试用例既包括自动测试脚本，也包括测试用例的手工实现过程。

测试用例的维护是一个不间断的过程，维护的主要内容包括下述几个方面：

1. 删除过时的测试用例

因为需求的改变等原因可能会使一个基线测试用例不再适合被测试系统，这些测试用例就会过时。例如，本项目原有的软件需求规定了软件应当实现对神经网络模型实例运行时使用的CPU数量的自定义功能，后来在项目部署过程中，发现该功能的意义不大，因此未实现该功能。对应地，该功能的测试用例也在维护过程中被删去。

1. 改进不受控制的测试用例

随着软件项目的进展，测试用例库中的用例会不断增加，其中会出现一些对输入或运行状态十分敏感的测试用例，这些测试不容易重复且结果难以控制，会影响回归测试的效率，需要进行改进，使其达到可重复和可控制的要求。

1. 增添新的测试用例

如果某个程序段、构件或关键的接口在现有的测试中没有被测试，那么应该开发新测试用例重新对其进行测试，并将新开发的测试用例合并到测试集中，通过对测试用例集的维护不仅改善了测试用例的可用性，同时还可以将一个基线测试用例库的效率保持在一个较高的级别上。

## 1.3 回归测试策略

回归测试是在开发人员执行了软件缺陷修复后，对软件进行重新测试的过程。回归测试主要包含两方面的考虑：a) 测试软件缺陷修复所做的改动是否修复了报告的软件缺陷；b) 测试软件缺陷修复所做的改动是否影响整体系统的预期行为。

本组在平衡效率和有效性的前提下，选择了部分的基线测试用例用于构造回归测试用例。测试者通过相关性分析，识别软件的修改情况并分析修改的影响，将回归测试局限于被改变的模块和它的接口上。在允许的条件下，回归测试应当尽可能地覆盖受到影响的部分。

对于本项目而言，表1总结了从项目第一次发布到执行回归测试前所发现的软件缺陷，以及为检查这些软件缺陷所采用的回归测试用例。

**表1 软件缺陷与对应的回归测试用例**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 软件缺陷名称 | 针对性回归测试用例 | 共用回归测试用例 |
| 1 | 注册邮箱的格式检查功能未正确执行 | 用例001，用例101 | 用例101 至 用例115  用例402，用例403，用例405 |
| 2 | 未登录用户越权访问用户数据 | 用例002，用例102，用例501 |
| 3 | 用户创建同名项目时的提示信息与预期不符 | 用例003，用例104 |
| 4 | 删除正在运行的实例时的提示信息与预期不符 | 用例011，用例113 |
| 5 | 模型配置中参数的数据类型错误 | 用例008，用例109 |
| 6 | 删除实例后配置文件与数据库信息不对等 | 用例009，用例113，用例502 |

## 1.4 回归测试的基本过程

回归测试的基本过程如图1所示：

**图1 回归测试的基本过程**

1. 维护基线测试用例集：从原有的基线测试用例集中，排除所有不再适用的测试用例，确定那些对新的软件版本依然有效的测试用例，以此建立一个新的基线测试用例集；
2. 挑选测试用例：针对软件缺陷的修复情况，从基线测试用例集中挑选与本次缺陷修复有关的测试用例，构成回归测试用例集；
3. 补充测试用例：构造测试用例用于直接测试本次缺陷修复过程新增的代码是否符合预期，将其加入回归测试用例集中，对于本次回归测试而言，用例501与用例502为补充测试用例，将在第3章进行详细叙述；
4. 执行回归测试用例集。

# 2 基线测试用例的回归测试结果

本章描述了对挑选的基线测试用例集进行的回归测试结果，共包括代码单元、功能需求、软件质量特征三个小节。

## 2.1 代码单元的回归测试

**表2 代码单元的回归测试结果**

| **测试用例** | **测试点** | **测试结果** | **测试状态** |
| --- | --- | --- | --- |
| 用例001 login/regist接口单元测试 | 数据缺失：当发送数据有缺失时，断言写为 self.assertEqual(resp["code"], 1005)时，执行结果为pass | 与预期相符 | 通过 |
| 数据格式错误：注册邮箱不符合邮箱格式时，断言写为 self.assertEqual(resp["code"], 1002)，执行结果为pass | 与预期相符 | 通过 |
| 账户名已存在：断言写为 self.assertEqual(resp["code"], 2002)，执行结果为pass | 与预期相符 | 通过 |
| 确认密码错误：注册输入的两个密码不一致时，断言写为 self.assertEqual(resp["code"], 2003)，执行结果为pass | 与预期相符 | 通过 |
| 注册成功：断言写为 self.assertEqual(resp["code"], 1000)，执行结果为pass | 与预期相符 | 通过 |
| 用例002 login/login接口单元测试 | 数据缺失：当发送数据有缺失时，断言写为 self.assertEqual(resp["code"], 1005)时，执行结果为pass | 与预期相符 | 通过 |
| 数据格式错误：输入邮箱不符合邮箱格式时，断言写为 self.assertEqual(resp["code"], 1002)，执行结果为pass | 与预期相符 | 通过 |
| 用户不存在：断言写为 self.assertEqual(resp["code"], 1007)，执行结果为pass | 与预期相符 | 通过 |
| 密码错误：输入密码与数据库对应的账户密码不一致时，断言写为 self.assertEqual(resp["code"], 2017)，执行结果为pass | 与预期相符 | 通过 |
| 登录成功：断言写为 self.assertEqual(resp["code"], 1000)，执行结果为pass | 与预期相符 | 通过 |
| 用例003 project/addPro接口单元测试 | 数据缺失：当发送数据有缺失时，断言写为 self.assertEqual(resp["code"], 1005)时，执行结果为pass | 与预期相符 | 通过 |
| 已存在同名项目：该用户已经有同名项目时，断言写为 self.assertEqual(resp["code"], 2004)，执行结果为pass | 与预期相符 | 通过 |
| 新建项目成功：断言写为 self.assertEqual(resp["code"], 1000)，执行结果为pass | 与预期相符 | 通过 |
| 用户未登录：断言写为 self.assertEqual(resp["code"], 1001)，执行结果为pass | 与预期相符 | 通过 |
| 用例008 model/editParam接口单元测试 | 数据缺失：当发送数据有缺失时，断言写为 self.assertEqual(resp["code"], 1005)时，执行结果为pass | 与预期相符 | 通过 |
| 模型不存在：断言写为 self.assertEqual(resp["code"], 2007)，执行结果为pass | 与预期相符 | 通过 |
| 用户未登录：断言写为 self.assertEqual(resp["code"], 1001)，执行结果为pass | 与预期相符 | 通过 |
| 模型已经部署，不能修改参数：断言写为 self.assertEqual(resp["code"], 2019)，执行结果为pass | 与预期相符 | 通过 |
| 修改成功：断言写为 self.assertEqual(resp["code"], 1000)，执行结果为pass | 与预期相符 | 通过 |
| 用例009 model/deleteModel接口单元测试 | 数据缺失：当发送数据有缺失时，断言写为 self.assertEqual(resp["code"], 1005)时，执行结果为pass | 与预期相符 | 通过 |
| 模型不存在：断言写为 self.assertEqual(resp["code"], 2007)，执行结果为pass | 与预期相符 | 通过 |
| 用户未登录：断言写为 self.assertEqual(resp["code"], 1001)，执行结果为pass | 与预期相符 | 通过 |
| 模型正在运行，不能删除模型：断言写为 self.assertEqual(resp["code"], 2013)，执行结果为pass | 与预期相符 | 通过 |
| 删除成功：断言写为 self.assertEqual(resp["code"], 1000)，执行结果为pass | 与预期相符 | 通过 |

## 2.2 功能需求的回归测试

**表3 功能需求的回归测试结果**

| **测试用例** | **测试点** | **测试结果** | **测试状态** |
| --- | --- | --- | --- |
| 用例101用户注册功能测试 | 用户是否能正常注册 | 可以正常注册 | 通过 |
| 用户注册信息要求是否能给出提示 | 可以正常给出提示 | 通过 |
| 用户注册的反馈是否能够正确通过页面显示 | 可以正常反馈 | 通过 |
| 表单数据是否能按照要求传递给后端 | 可以按照要求传递给后端 | 通过 |
| 用例102 用户登录功能测试 | 用户是否能正常登录 | 正常登录 | 通过 |
| 页面是否能准确反馈用户登录信息匹配问题 | 对于密码/用户名错误给出相应的提示 | 通过 |
| 用例103 用户登出功能测试 | 用户是否能正常登出 | 正常登出 | 通过 |
| 登出后的信息是否进行注销 | 登出后，无法对系统进行操作 | 通过 |
| 用例104 用户新建项目功能测试 | 用户是否能新建项目 | 可以正常建立项目 | 通过 |
| 项目新建后在页面中的展现是否清晰 | 可以正常展示项目 | 通过 |
| 用例105 用户查看项目功能测试 | 是否能够查看用户已存在的项目 | 能够正常查看用户已存在的项目 | 通过 |
| 用例106 用户删除项目功能测试 | 用户是否能删除个人项目中的项目 | 能够正常删除个人项目中的项目 | 通过 |
| 用例107用户更新项目功能 | 用户是否能够更新项目的具体信息 | 用户是否能够更新更新 | 通过 |
| 项目下所有的模型和实例对应的相关信息是否能够同步更新 | 项目下所有的模型和实例对应的相关信息能够同步更新 | 通过 |
| 用例108在项目中导入模型功能测试 | 是否能够成功在对应项目下导入模型 | 能够成功在对应项目下导入模型 | 通过 |
| 是否能够导入多个模型，相同模型能够避免重复导入 | 能够导入多个模型，相同模型能够避免重复导入 | 通过 |
| 用例109模型部署参数设置功能测试 | 是否能够按照用户需求对模型的参数进行合理配置 | 能够按照用户需求对模型的参数进行合理配置 | 通过 |
| 是否与服务器部署时的参数保持一致 | 部署的参数能够保持一致 | 通过 |
| 用例110查看项目下所有模型及其配置的功能测试 | 能否查看项目下特定模型的配置信息、参数信息、版本号 | 可以查看项目下所有模型及其配置 | 通过 |
| 能够查看的模型是否包含项目下的所有模型文件 | 模型包含项目下的所有模型文件 | 通过 |
| 用例 111 删除项目下模型的功能测试 | 是否能够删除项目下的指定模型 | 能够删除项目下的指定模型 | 通过 |
| 用例112启动实例功能测试 | 门户系统是否存储正确文件所在服务器的位置 | 门户系统正确存储文件所在服务器的位置 | 通过 |
| 服务器启动对应配置文件参数的机器学习运行加载模型 | 服务器启动对应配置文件参数的机器学习运行加载模型 | 通过 |
| 端口分配是否合理 | 端口分配没有冲突 | 通过 |
| 用例 113 删除实例功能测试 | 删除实例后是否不再消耗服务器内存及计算资源 | 删除实例后不再消耗服务器内存及计算资源 | 通过 |
| 无需响应用户的请求，端口释放 | 无需响应用户的请求，端口释放 | 通过 |
| 用例 114 暂停实例功能测试 | 是否能够停止该实例对外提供服务并保留该实例所在远端服务器的运行 | 能够停止该实例对外提供服务并保留该实例所在远端服务器的运行 | 通过 |
| 用例 115 恢复实例功能测试 | 是否能够恢复该实例对外提供的服务，实例可以响应外界发送过来的请求 | 能够恢复该实例对外提供的服务，实例可以响应外界发送过来的请求 | 通过 |

## 2.3 软件质量特征的回归测试

**表4 软件质量特征的回归测试结果**

| **测试用例** | **测试点** | **测试结果** | **测试状态** |
| --- | --- | --- | --- |
| 用例402 特殊符号容错性测试 | 用户在注册、登录、管理项目时输入的特殊符号是否会引发系统异常 | 用户输入的特殊符号没有引发系统异常 | 通过 |
| 用例403 超长字符串容错性测试 | 用户在注册、登录、管理项目时输入的超长字符串是否会引发系统异常，系统是否能返回长度超过限制的错误提示 | 用户输入的超长字符串没有引发系统异常，系统能给出字符串长度超过限制的提示 | 通过 |
| 用例405 SQL注入攻击测试 | 恶意用户是否能通过各个文本输入框进行SQL注入攻击 | SQL注入攻击的尝试失败，构建的攻击语句没有被系统执行 | 通过 |

# 3 针对评审意见新增的测试用例

由于在测试需求设计阶段所构造的测试用例未能发现所有的软件缺陷，根据项目内部自测与评审人员反馈的问题，本次测试过程补充了两个测试用例，它们分别是用例501 未登录用户越权访问用户数据的安全测试，以及用例502 配置文件与数据库信息同步动作的测试。

## 3.1 用例501 未登录用户越权访问用户数据的安全测试

**表5 用例501 未登录用户越权访问用户数据的安全测试**

|  |  |  |
| --- | --- | --- |
| **名称** | 未登录用户越权访问用户数据的安全测试 | |
| **简要描述** | 本测试主要用于检查未登录用户通过浏览器地址越权访问其他用户数据的问题是否被正确修复 | |
| **前提和约束** | 服务器正常运行，客户机网络连接状况良好，且装有浏览器 | |
| **测试方法** | 黑盒测试 | |
| **测试步骤** | **1** | 新建项目test1，点击查看项目，记录当前的URL |
| **2** | 注销用户 |
| **3** | 在浏览器地址栏中输入步骤1记录的URL，访问该地址 |
| **预期结果** | 浏览器阻止未登录用户访问用户数据，浏览器返回到登录界面 | |
| **测试结果** | 测试结果与预期相符 | |
| **测试结论** | 该功能通过测试 | |
| **测试人员** | 崔昕宇 | |
| **测试时间** | 2020/5/24 | |

## 3.2 用例502 配置文件与数据库信息同步动作的测试

**表6 用例502 配置文件与数据库信息同步动作的测试**

|  |  |  |
| --- | --- | --- |
| **名称** | 配置文件与数据库信息同步动作的测试 | |
| **简要描述** | 本测试主要用于检查删除实例后配置文件与数据库信息不对等的问题是否被正确修复 | |
| **前提和约束** | 服务器正常运行，客户机网络连接状况良好，且装有浏览器 | |
| **测试方法** | 黑盒测试 | |
| **测试步骤** | **1** | 修改模型参数，并在查看模型详情界面点击“启动实例”； |
| **2** | 点击暂停实例使实例可以被删除； |
| **3** | 点击删除实例，跳转到设置部署参数界面； |
| **4** | 确认设置参数界面中，各参数都被清空。 |
| **预期结果** | 在用户没有对参数修改的情况下，再次点击“启动实例”，应当按照系统的默认参数运行实例 | |
| **测试结果** | 测试结果与预期相符 | |
| **测试结论** | 该功能通过测试 | |
| **测试人员** | 崔昕宇 | |
| **测试时间** | 2020/5/24 | |

# 4 总结与反思

本文档描述了C组开发项目 ***基于Flask的深度学习自动化部署系统*** 的回归测试概述及测试结果，通过回归测试，本组开发人员确认了项目第一次发布之后所发现的各个软件缺陷均已被修复，且对软件的修改过程没有引入新的问题。

反思在测试需求设计阶段所构造的测试用例未能完全覆盖程序执行情况的原因，不难发现：其一，软件需求未能覆盖所有的异常情况，开发人员在设计软件需求时，仅依照自身经验考虑了可能的异常情况，但实际应用中出现的异常情况往往难以被完全覆盖，测试用例501就是为了补充约定对异常情况的应对方式；其二，仅根据软件需求构造的测试用例只考虑了执行单一功能的情景，未考虑诸多功能叠加时的相互作用，测试用例502即补充约定了修改参数、删除实例、启动实例三个功能叠加时的软件行为。

# 参考文献

1. 邹欣.构建之法[M].北京.人民邮电出版社,2015:264-296
2. 软件工程测试方法[EB/OL]. https://blog.csdn.net/fightfaith/article/details/47858101
3. 软件测试之回归测试[EB/OL]. https://blog.csdn.net/zhusongziye/article/details/80383878